VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Matej Kubinec

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Matej Kubinec**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: SDE - Software Solutions s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Eliška Ochodková, Ph.D.**

Konzultant bakalářské práce: RNDr. Jaroslav Žáček, Ph.D.

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB – TU Ostrava.

V Ostravě 20.dubna 2018

SDE Software Solutions s.r.o.
Dornych 60, 617 00 Brno
IČ: 253 09 978
tel.: 545 211 280
(4)

Poďakovanie

Na tomto mieste by som sa rád poďakoval RNDr. Eliške Ochodkovej, Ph.D za vedenie tejto práce. Ďalej by som sa chcel poďakovať RNDr. Jaroslavovi Žáčkovi Ph.D a Ondřejovi Patočkovi za konzultácie a pomoc zo strany SDE Software Solutions. Moje poďakovanie patrí aj celému kolektívu firmy SDE Software Solutions za vytvorenie príjemného prostredia pre prácu.

Abstrakt

Tento dokument popisuje priebeh mojej odbornej bakalárskej praxe vo firme SDE Software Solutions. Firma sa zaraďuje medzi outsourcingové firmy a zameriava sa na americký trh. Vo firme som bol zaradený v tíme, ktorý spolupracuje so spoločnosťou Measurement Incorporated, ktorá ponúka vzdelávacie služby. Podieľal som sa na testovaní novej platformy pre online testovanie OASys ako aj na integrácii produktu Pegwriting s SSO službou Clever.

Kľúčové slová: Selenium, C# Specflow, BDD, testovanie softvéru, testovací scenár, automatizované testovanie

Abstract

This document describes my bachelor's practice at SDE Software Solutions. The company belongs to outsourcing companies and focuses on the American market. Within the company I was assigned to a team, which cooperates with Measurement Incorporated, a company which provides educational services. I have been involved in the testing of a new platform for online testing Oasys as well as in Pegwriting's integration with SSO service Clever.

Key Words: Selenium, C#, Specflow, BDD, software testing, test case, test automation

Obsah

Seznam použitých zkratek a symbolů

US	–	User Story
UI	–	User Interface
VSTS	–	Visual Studio Team Services
CI	–	Continuous Integration
MI	–	Measurement Incorporated
API	–	Application Programming Interface
CSV	–	Comma-separated values
SQL	–	Structured Query Language
SSO	–	Single Sign On system
BDD	–	Behavior Driven Development
AES	–	Automated Essay Scoring
PO	–	Product Owner

Zoznam obrázkov

Zoznam tabuliek

Seznam výpisů zdrojového kódu

1 Úvod

Tento dokument popisuje priebeh mojej odbornej praxe v rámci výuky na Technickej Univerzite Ostrava vo firme SDE.

V prvej časti dokumentu je bližšie priblížená firma SDE Software Solutions, v ktorej som vykonával odbornú prax. Priblížená je aj firma Measurement Incorporated s ktorou firma SDE Software Solutions spolupracuje a na ktorej projektoch som pracoval ako tester. V nasledujúcej kapitole sú popísané úlohy, na ktorých som v priebehu odbornej praxe pracoval. Riešenie a testovanie daných úloh je popísané v ďalšej kapitole. Posledné kapitoly dokumentu sa zaoberajú teoretickými aj praktickými znalosťami nadobudnutými počas štúdia ako aj vedomosťami, ktoré mi v priebehu praxe chýbali. V samotnom závere zhodnocujem celkový priebeh odbornej praxe.

2 Popis odborného zamerania firmy

2.1 Popis spoločností

2.1.1 SDE

Spoločnosť SDE Software Solutions vznikla v roku 1995 v Brne. Bola založená americkými obchodníkmi a zaoberá sa outsourcingom[1]. To znamená, že zamestnáva v Českej republike programátorov a prepožičiava ich americkým IT spoločnostiam, ktorým pomáhajú s vývojom produktov. Medzi projekty na ktorých firma pracuje patrí napríklad informačný systém pre pacientov a doktorov, systém automatickej ochrany pred únikom vody FloLogic atď. V rámci projektov, na ktorých firma spolupracuje, sa používa široké spektrum programovacích jazykov a technológií ako C++, Java, C#, Javascript a frameworky ako Angular a React, Ruby on Rails a iné.

Obr. 1: SDE Software Solutions

2.1.2 Measurement Incorporated

Measurement Incorporated je firma z Durhamu v Severnej Karolíne zaoberajúca sa sprostredkovaním vzdelávacích služieb. Zamestnáva takmer 400 zamestnancov naprieč 14 pobočkami v 9 štátoch USA. Medzi ich produkty patria PEG, PEG Writing, PEG Writing Scholar a MIPS. Tieto systémy slúžia na automatizované hodnotenie slohových prác študentov ale aj na elektronické testovanie študentov.

Obr. 2: Measurement Incorporated

2.2 Popis odborného zamerania študenta

Vo firme som pracoval ako tester. Náplňou mojej práce bolo zaručenie splnenia akceptačných kritérií, manuálne testovanie ako aj automatizácia testovacích scenárov. V priebehu mojej praxe som pracoval na viacerých produktoch firmy Measurement Incorporated ako, je administratívny portál pre novú platformu OASys, ako aj na staršom produkte PEG Writing.

[1] Outsourcing znamená, že firma vyčlení rôzne podporné a vedľajšie činnosti a zverí ich inej spoločnosti, čiže subkontraktovi, špecializovanému na príslušnú činnosť.

2.3 Workflow v Measurement Inc.

Riadenie vývoja zodpovedalo procesnému frameworku scrum. Súčasťou 9 členného tímu bol scrum master, product owner, 5 programátorov a 2 testery. Časť tímu bola z firmy SDE (scrum master, 2 programátori a tester) a časť tímu z firmy Measurement Inc. Na základe rizika bol sprint[2] týždňový (napr. keď sme prechádzali na starší produkt a museli sme sa s ním zoznámiť.) alebo 2-týždňový. Súčasťou sprintov bol aj denný stand-up, na ktorom každý člen tímu informoval ostatných na čom práve pracuje, či nenarazil na nejaký problé, alebo potreboval vysvetliť nejaké nejasnosti. Dva dni v týždni sme mávali aj meeting, na ktorom sa bližšie špecifikovali user story, ktoré budú súčasťou nasledujúcej iterácie. Tohto meetingu sa zúčastňoval každý člen tímu a spoločne sa špecifikovali user story[3], taktiež sa určovali story pointy[4] danej úlohy. Sprint začína sprint planningom, na ktorom sa priradia úlohy konkrétnym ľudom, určí sa ich časový odhad na implementáciu a otestovanie. V rámci plánovania prebieha aj retrospektíva za predošlú iteráciu. Dané US boli zaznamenávané v rámci VSTS (Visual Studio Team Services), v ktorom sú zaznamenávané zápisy z retrospektív a dokumentácia k projektom. Vo VSTS sú definované aj jednotlivé kroky CI procesov. Na komunikáciu používame Slack, poprípade Skype medzi Ostravskou časťou tímu.

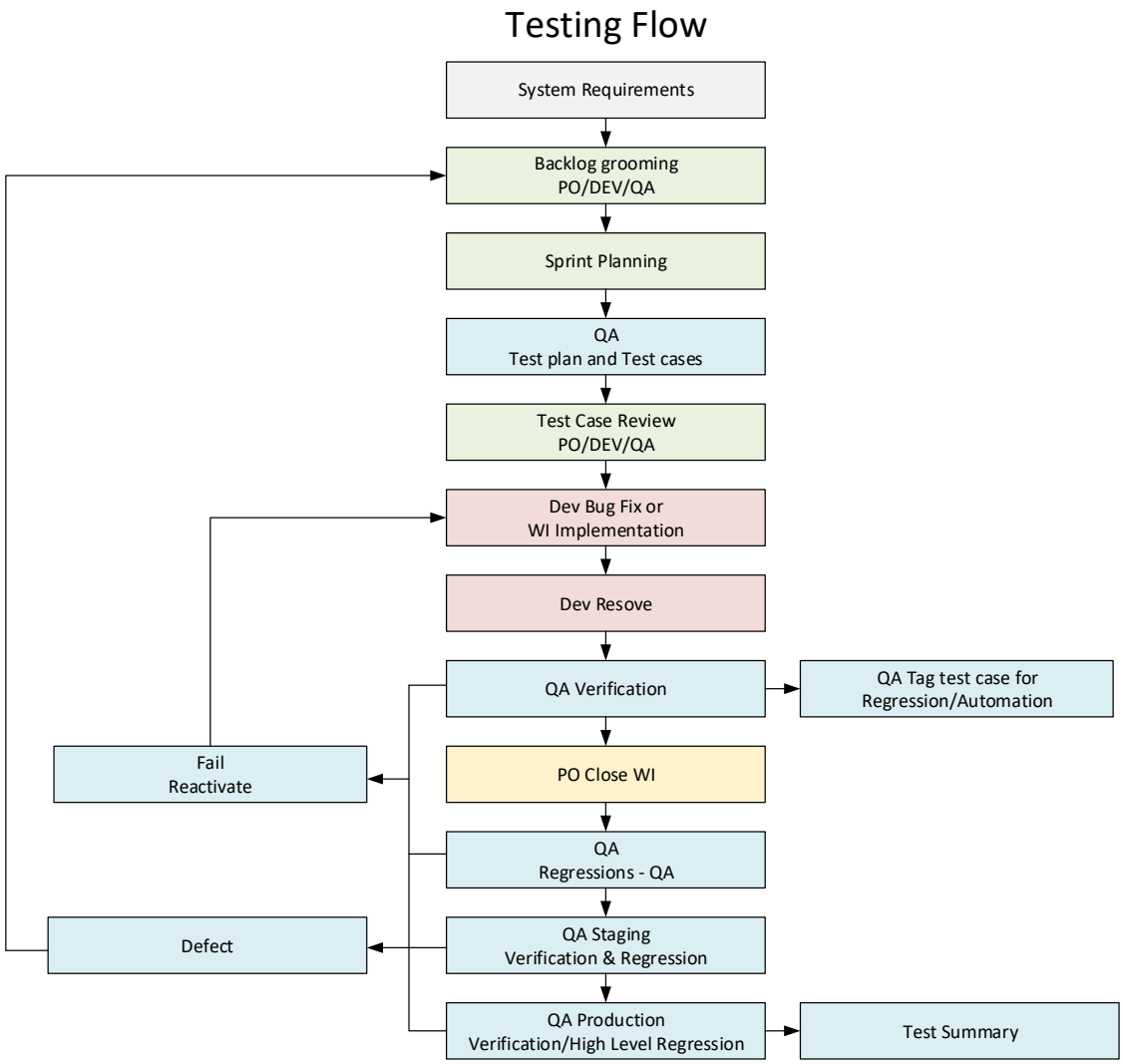
2.4 Testovací proces

Testovací proces je unifikovaný naprieč viacerými projektami a tímami MI, zodpovedá obrázku 3. Pri zaradení do sprintu je pre danú úlohu zostavený testovací prípad, ktorý na základe akceptačných kritérií obsahuje kroky, ktoré budú vykonávané pri testovaní. Po implementácií a nasadení na testovací server je úloha pripravená na otestovanie. Pokiaľ implementácia spĺňa akceptačné kritéria je úloha uzavretá a označená na regresiu a automatizáciu (pokiaľ sa jedná o úlohu ktorá je na to vhodná). Ak úloha nespĺňa akceptačné požiadavky je vypísaný defekt, ktorý skontroluje PO a určí, či je chyba natoľko závažná, aby bola opravená v priebehu aktuálne prebiehajúceho sprintu, alebo je možné ju odložiť do neskorších sprintov. Pokiaľ má byť chyba opravená počas práve prebiehajúceho sprintu je vrátená vývojárovi na opravenie a celý proces sa opakuje. Pri regresnom testovaní sa využije testovací prípad úlohy, aby sa skontrolovalo, či si neobjavili nedostatky počas implementácie iných úloh.

[2]Sprint je jedna iterácia vývoja, na konci ktorej je dodaná zákazníkovi hodnota.

[3]User story je neformálny popis funkcie systému z pohľadu užívateľ daného systému.

[4]Story point je číslo, ktoré na základe rizika, časovej náročnosti implementácie a testovania reprezentuje úsilie na dokončenie úlohy.



Obr. 3: Testovací proces Measurement Incorporated

2.5 Writing Websites

Produkty Pegwriting, Pegwriting Scholar, Utah Compose a Writing Practice Program sú webové aplikácie so spoločnou funkcionalitou, ktoré slúžia na vylepšenie písacích schopností žiakov. Na hodnotenie prác sa používa AES engine PEG, ktorý automaticky ohodnotí dané práce a ponúkne spätnú väzbu. V rámci týchto stránok sme implementovali integráciu Pegwriting s SSO službou Clever[5] [7].

[5]Clever je platforma, ktorá umožňuje jednotný prístup k viacerým vzdelávacím aplikáciám.

2.6 OASys administratívny portál

Administratívny portál OASys je nový produkt Measurement Incorporated, ktorý slúži na administratívu užívateľov a testov žiakov. Je jednou z častí novej platformy OASys. V rámci administratívneho portálu sa využívajú aktuálne technológie, .NET Core 2 na backende a na frontende sa používa javascriptový framework React spolu so staticky typovaným jazykom typescript. V rámci testovacích technológií sa pre automatizáciu akceptačných testov používa Selenium [1] a Specflow [2], ktorý slúži na prepojenie business požiadavkov na .NET Core. Využíva jazyk Gherkin [3], ktorý umožňuje písanie akceptačných testov, ktoré sú čitateľné aj pre ľudí bez znalostí daných technológii.

S automatizáciou akceptačných testov pomocou Selenia mám skúsenosti už z predošlého projektu na ktorom som pracoval, ale Specflow a BDD prístup boli pre mňa nové. Bohužiaľ Specflow oficiálne nepodporuje .Net Core a bolo nutné použiť knižnicu, ktorá zabezpečila kompiláciu *feature* súborov. Táto nekompatibilita spôsobuje aj niektoré iné problémy, akými sú nemožnosť aktualizovať balíčky na najaktuálnejšiu verziu Specflow a nemožnosť spúšťať automatizované testy na inom operačnom systéme ako Windows.

3 Zoznam zadaných úloh

3.1 Nastavenie testovacieho prostredia pre automatizovane testovanie

Cieľom danej úlohy bolo nastaviť prostredie pre automatizované testy, ktoré by zachytili chyby už v implementovaných častiach softvéru. Testy by mali byť spúšťané buď každý pracovný deň v nočných hodinách, alebo po každom zostavení aplikácie. Selenium testy budú spúšťané na najaktuálnejšej verzii prehliadača Google Chrome v službe Browserstack[6] [8].

3.2 Triedenia dát v tabuľke študentov

Cieľom úlohy bolo umožniť užívateľom administratívneho portálu triedenia dát v tabuľke podľa každého stĺpca, aby mohli jednoducho nájsť študentov, pokiaľ o nich už nejaké informácie vedia, ako sú meno študenta, užívateľské meno a podobne.

Predvolené triedenie bude podľa priezviska študenta. Každý stĺpec bude možné triediť zostupne aj vzostupne. Triedenie by malo pretrvať počas relácie užívateľa, triedenie sa nezmení pokiaľ užívateľ prejde na inú stránku. Triedenie sa vráti do pôvodného stavu v prípade, ak užívateľ obnoví stránku alebo sa odhlási.

Stĺpce by mali indikovať, že ich je možné zotriediť a pokiaľ už zotriedené sú, tak by mali ukazovať smer triedenia. Dizajn by mal zodpovedať návrhu ako na obrázku 4.

[6] Browserstack je služba, ktorá umožňuje spúšťať automatizované UI testy na rôznych zariadeniach a webových prehliadačoch.

Student	Username
Ballou, Nellie	nballou@measinc.com
Bogner, Virginia	vbogner@dcps.gbcc.12k.edu
Delgado, Lisa	lisa.delgado@cmpd.westcharlotte...
Lopez, Rachel	rachel.g.lopez@cmpd.eastcharlotte...
Paddock, Theresa	tpaddock@yahoo.com
Raines, Margaret	mraines@measinc.com
Sanders, Eric	ericasanders@gmail.com
Stevenson, Ethel	swetherell@measinc.com
Paddock, Theresa	tpaddock@yahoo.com
Raines, Margaret	mraines@measinc.com
Stevenson, Ethel	e.stevensonl@measinc.com
Paddock, Theresa	tpaddock@yahoo.com

Obr. 4: Návrh hlavičky tabuliek a triedenia dát v tabuľke

3.3 Filtrovanie dát v tabuľke

Zámerom tejto úlohy bolo zabezpečiť užívateľom vykonávať filtrovanie dát v tabuľkách. Filtrovať sa mal každý stĺpec tabuľky. Vstupné polia na filtrovanie by mali byť v prvotnom stave skryté, zobraziť by sa mali až po stlačení tlačidla. Po stlačení sa zobrazia pod každou hlavičkou vstupné polia, do ktorých bude možné zadať reťazec na filtrovanie. Pri prejdení ponad tlačidlo sa zobrazí nápoveda *Filter Rows*. Ak je filtrovanie aktívne, tak pozadie tlačidlá je zvýraznené. Filtrovanie sa vykonáva hneď po zadaní znakov do vstupného poľa, filtruje sa na základe celého reťazca a filtrovať je možné aj na základe viacerých stĺpcov. Pri skrytí vstupných polí sa filtrovanie vráti do pôvodného stavu. Pokiaľ žiadne dáta nezodpovedajú filtrovaným hodnotám, zobrazí sa správa, ktorá to užívateľovi oznámi. Filtrovanie by malo pretrvať počas navigácie v aplikácii. Pri obnovení prehliadača sa filtre vyprázdnia.

3.4 Pridávanie škôl, kurzov, učiteľov a študentov cez import

Počas synchronizácie s Cleverom bolo nutné pridať do databáze školy, učiteľov, žiakov a kurzy, ktoré nám sprístupnia. Používatelia Clevera, ktorí budú využívať Pegwriting, nebudú mať možnosť pridávať školy, učiteľov a študentov. V databáze sa nesmú počas importu vytvoriť duplikáty.

3.5 Pridávanie študentov do kurzu a udelenie licencií počas importu

Ďalšou úlohou v rámci synchronizácie s Cleverom bolo priradenie študentov do kurzov. S priradzovaním súvisel systém priraďovania licencií. Každá licencia umožňuje jednému študentovi využívať služieb Pegwriting, to znamená, že môže byť zároveň vo viacerých kurzoch, ale spotrebúva iba jednu licenciu. Licencie sa udeľujú na úrovni školy. Pokiaľ škola nebude mať pri importe dostatočné množstvo licencií, tak študenti, na ktorých neostane licencia, nebudú pridaní do kurzu a v importe sa objaví správa, že študenti neboli pridaní. Pokiaľ študent zmení kurz a je importovaný znova, tak je odstránený z predošlého kurzu a je priradený do nového. Ak študent zmení školu jeho slohové práce musia zostať zobraziteľné.

4 Postup riešenia zadaných úloh

4.1 Nastavenie testovacieho prostredia pre automatizované testovanie

Moja prvá úloha v rámci spolupráce s firmou MI slúžila primárne na zoznámenie sa s projektom, danými technológiami a testovacím prostredím. Po analýze možností na interakciu s webovými stránkami som sa rozhodoval medzi frameworkami Selenium, WebdriverIO [6] a Cypress [5]. Selenium je najstarším a najpoužívanejším frameworkom pre automatizáciu, je open source a existujú pre neho previazania a knižnice na rôzne technológie, platformy a jazyky. WebdriverIO je implementáciu webdriver rozhrania Selenia v jazyku javascript pre nodeJS. Cypress je all-in-one testovací framework, ktorý nevyužíva na pozadí Selenium, je postavený na javascripte a je vhodný pre moderné javascriptové frameworky ako Angular alebo React. Nakoniec som sa rozhodol pre framework Selenium kvôli dobrej podpore .Net frameworku a pre spoluprácu s BDD frameworkom Specflow pre naviazanie testovacích scenárov na príkazy Selenia.

Vytvoril som základnú štruktúru testovacieho projektu, ktorý sa skladal z hlavného projektu, ktorý obsahoval Specflow scenáre a naviazania na dané kroky scenára a knižnice, ktorá obsahuje implementáciu stránok. Začal som tým, že som si prečítal dokumentáciu Specflow-u a oboznámil som sa so štruktúrou a syntaxou jazyka gherkin.

Jednoduchý scenár v jazyku gherkin vyzerá následovne. Každý scenár sa skladá z troch krokov:

1. **Given** - cieľom je dostať systém do známeho stavu (príprava na interakciu)
2. **When** - popisuje akciu, ktorú užívateľ vykoná (interakcia so systémom)
3. **Then** - popisuje výsledok akcie (overenie výsledku)

And a **But** používame ak máme niekoľko krokov rovnakého typu, robia zápis čitateľnejším.

Po naštudovaní si danej syntaxe som pridal testovacie scenáre pre prihlasovanie uživateľa. Bolo treba zachytiť úspešné prihlásenie, presmerovanie užívateľa, neúspešné prihlásenie, ako aj chybové správy pri vynechaní prihlasovacieho mena, hesla a pri zadaní nesprávnych údajov. Pri tomto scenári som využil aj ďalšiu konštrukciu jazyka gherkin **Examples**, ktorá umožňuje využiť testovací scenár pre viacero vstupných dát.

Pre vlastnú interakciu s webovou stránkou som vytvoril triedu na základe návrhového vzoru Page Object [4]. Pre každý element na prihlasovacej stránke som si vytvoril premennú, ktorá ho reprezentovala. Na prihlasovanie užívateľa som si vytvoril metódu, ktorá vykonala danú sériu krokov. Diagram triedy môžeme znázorniť ako na obrázku 5. Vlastné previazanie na Specflow scenárov vyzerá ako na výpise 2.

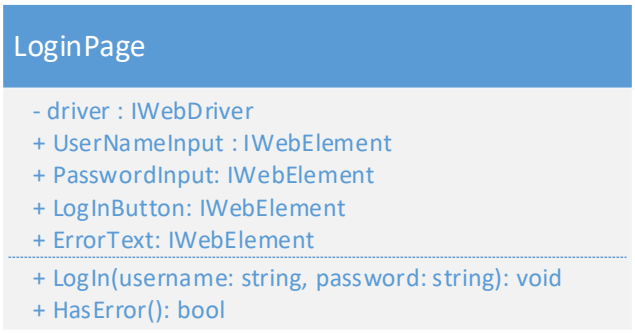
```
Feature: LogIn
    In order to use Administration
    As a user
    I want to be able to log in

Scenario: Log in fails
    Given I have navigated to the Login page
    When I log in with invalid credentials
    Then I should see an error message

Scenario: Log in succeeds
    Given I have navigated to the Login page
    When I log in with valid credentials
    Then I should not see an error message

Scenario Outline: Error messages
    Given I have navigated to the Login page
    When I log in with username "<username>" and password "<password>"
    Then I should see an error message
    And Error message should be <error message>
    Examples:
    | username | password | error message                                                |
    |          | password | The Username field is required.                              |
    | username |          | The Password field is required.                              |
    | username | password | Your username or password is incorrect. Please try again. |
```

Výpis 1: Testovacie scenáre prihlasovania užívateľa



Obr. 5: Diagram Login Page Object-u

Ďalším krokom bolo spúšťanie daných testovacích scenárov na službe Browserstack. Pri prvom pokuse o spustení testov som zistil, že aktuálna verzia Selenia má defekt na platforme .Net Core, ktorá znemožňuje vykonanie daných scenárov na vzdialenom serveri. Dočasným riešením bolo skopírovanie danej triedy, ktorá vyhadzovala danú výnimku a zakomentovanie 2 riadkov, ktoré ju spôsobovali. Po tom ako som nastavil, aby sa využívala daná upravená trieda, sa scenáre spúšťali korektne.

Posledným krokom bolo zaistiť, aby sa testy spúšťali v ranných hodinách. Vo VSTS som špecifikoval vlastnú konfiguráciu zostavenia, ktorá sa mala vykonávať vždy o 7:00 ráno. Po zostavení sa dané testovacie scenáre vykonali na testovacom serveri. Po dokončení bol výsledok testovania

```
[Given(@"I have navigated to the Login page")]
public void GivenIHaveNavigatedToTheLoginPage()
{
    _driver.Url = TestData.Data["url"].ToString();
}

[When(@"I log in with username ""(.*)"" and password ""(.*)""")]
public void WhenILogInWithUsernameAndPassword(string p0, string p1)
{
    _loginPage.LogIn(p0, p1);
}

[Then(@"I should see an error message")]
public void ThenIShouldSeeTheError()
{
    Assert.IsTrue(_loginPage.HasError);
}
```

Výpis 2: Previazanie Specflow na Selenium

dostupný v súhrne, zobrazoval počet úspešne a neúspešne prevedených testov. Dostupný bol aj výpis z konzoly, ktorý umožnil zistiť na ktorom kroku daný testovací prípad zlyhal.

Nastavenie testovacieho prostredia spolu s analýzou možností a následnou implementáciou automatizovaných testov mi zabralo 3 pracovné dni.

4.2 Testovanie triedenia dát v tabuľke študentov

Prvým krokom v testovaní danej implementácie bolo zostavenie testovacieho scenára, podľa ktorého bude daná implementácia testovaná. Výsledný testovací scenár zodpovedá tabuľke 1. Po implementácii a nasadení na testovací som triedenie tabuliek overil na základe testovacieho scenára na všetkých podporovaných zariadeniach. Dizajn hlavičiek som overil podľa návrhu. Po overení funkčnosti a dizajnu som začal s automatizáciou daného testovacieho scenára.

Prvým krokom v automatizáciu je vytvorenie Specflow feature-y s testovacími scenármi v jazyku gherkin. Testovacie scenára súvisia priamo už s danými implementovanými tabuľkami. Pre interakciu s danou tabuľkou som vytvoril triedu na základe návrhového vzoru Page Object. Pre elementy na stránke ako tlačidlá pre ďalšiu/predošlú stránku, vstup čísla stránky, select počtu záznamov. Pre interakciu so stĺpcami som si vytvoril metódu, ktorá na základe mena stĺpca klikne na daný stĺpec, ako aj metódu ktorá prejde záznamy v danom stĺpci tabuľky a overí, či sú zoradené vzostupne alebo zostupne na základe vstupných parametrov. Táto úloha bola ohodnotená 5 story point-mi, pričom väčšinu z týchto bodov tvorilo úsilie na implementáciu. Otestovanie implementácie na podporovaných zariadenia a vytvorenie testovacieho prípadu mi zabralo približne 8 hodín.

Tabuľka 1: Testovací scenár - Triedenie tabuliek

Krok	**Akcia**	**Očakávaný výsledok**
1	Pokiaľ som užívateľom administratívneho portálu a zobrazujem tabuľku študentov alebo užívateľov	
		Tak tabuľka je zoradená podľa priezviska Zotriediteľné stĺpce indikujú, že je ich možné zoradiť Nezotriedené stĺpce majú šípky do obidvoch smerov
2	Keď užívateľ klikne na hlavičku stĺpca	
		Tak stĺpec bude indikovať, že je zoradený Hlavička bude používať korektnú ikonu podľa smeru triedenia Tabuľka bude zoradená podľa daného stĺpca
3	Keď užívateľ klikne na hlavičku stĺpca, ktorý nie je zoradený a obsahuje textové dáta	
		Potom tabuľka zoradená od A do Z
4	Keď užívateľ klikne na tú istú hlavičku znova	
		Potom je tabuľka zoradená od Z do A
5	Keď užívateľ klikne na hlavičku stĺpca, ktorý nie je zoradený a obsahuje numerické dáta	
		Potom je tabuľka zoradená podľa daného stĺpca od najmenšieho čísla po najväčšie.
6	Keď užívateľ klikne na tú istú hlavičku znova	
		Potom je tabuľka zoradená od najväčšieho čísla po najmenšie
7	Keď užívateľ klikne na hlavičku stĺpca, ktorý nie je zoradený a obsahuje dátumy alebo časové záznamy	
		Potom je tabuľka zoradená podľa daného stĺpca od najstaršieho záznamu po najaktuálnejší.
8	Keď užívateľ zmení stránku a vráti sa späť na stránku s tabuľkou	
		Tak zoradenie tabuľky pretrvá
9	Keď užívateľ obnoví danú stránku	
		Zoradenie tabuľky sa vráti späť do pôvodného stavu

4.3 Automatizácia akceptačných testov pre filtrovanie dát v tabuľke

Pri automatizácii testov som sa riadil testovacím prípadom, ako aj akceptačnými kritériami v user story pre filtrovanie dát v tabuľke. Začal som si vypísaním stavov, ktoré musíme v rámci testovania zachytiť a otestovať. Interakciu užívateľa s filtrami tabuľky môžeme zapísať nasledovne:

- **Uživateľ sa navigoval na stránku s tabuľkou** - Predvolený stav - filtre sú prázdne, nie sú viditeľné na UI
- **Uživateľ klikol na tlačidlo zobrazenia filtrov** - filtre sa zobrazia a sú viditeľné
- **Uživateľ zadal text do vstupného poľa filtra** - tabuľka je vyfiltrovaná na základe daného reťazca
- **Uživateľ zadal text do ďalšieho vstupného poľa** - tabuľka je vyfiltrovaná podľa obidvoch reťazcov
- **Užívateľ zadal reťazec, ktorý sa v tabuľke nenachádza** - tabuľka je prázdna, užívateľovi je zobrazená správa, že také dáta sa v tabuľke nenachádzajú.
- **Užívateľ prepol na inú stránku a vrátil sa späť na stránku s tabuľkou** - tabuľka je stále filtrovaná
- **Užívateľ obnovil stránku** - tabuľka už nie je naďalej filtrovaná

Pokiaľ máme filter, ktorý filtruje na základe numerických hodnôt, máme tieto prípady, ktoré treba otestovať:

- Do vstupného poľa je znemožnené užívateľovi zadať nenumerickú hodnotu
- **Užívateľ zadá nekorektnú hodnotu (záporné číslo pri počte testov)** - Užívateľovi sa zobrazí chybová správa.
- **Užívateľ zadá hodnotu ktorá sa v tabuľke nachádza** - tabuľka je vyfiltrovaná na základe zadanej hodnoty
- **Užívateľ zadá hodnotu ktorá sa v tabuľke nenachádza** - tabuľka bude prázdna a užívateľovi sa zobrazí, že zadaná hodnota ta nenašla.

Pokiaľ máme filter, ktorý filtruje na základe reťazca, máme tieto prípady ktoré musíme zachytiť:

- **Užívateľ zadal len medzery** - tabuľka nie je filtrovaná
- **Užívateľ zadal reťazec** - tabuľka je filtrovaná na základe daného reťazca
- **Užívateľ zadal reťazec s medzerami na začiatku/na konci** - tabuľka je filtrovaná na základe reťazce, medzery sa ignorujú
- **Užívateľ zadal reťazec s medzerami uprostred** - tabuľka je filtrovaná na základe reťazca aj s medzerou

```
Scenario: Filters are hidden by default
   Given I am logged in as teacher
   When I am viewing the students grid in its default state
   Then The filters should be hidden

Scenario: Show filters
   Given I am logged in as teacher
   When I am viewing the students grid in its default state
   And I select the filter icon
   Then The filters should be visible

Scenario: Hide filters
   Given I am logged in as teacher
   When I am viewing the students grid in its default state
   And I select the filter icon
   And I select the filter icon
   Then The filters should be hidden

Scenario: Negative numbers
   Given I am logged in as test coordinator
   When I am viewing the users grid in its default state
   And I select the filter icon
   And I type in -1 box under Tests Scheduled
   Then There should be an error message
```

Výpis 3: Úryvok Specflow scenárov

Z týchto prípadov som zostavil Specflow scenáre, časť z nich môžeme vidieť na výpise 3. Následne som vytvoril na Page Object-e, ktorý zaobstaráva interakciu s tabuľkami, funkcie, ktoré sa starali o užívateľskú interakciu s filtrami tabuliek. Na to, aby som podľa názvu hlavičky mohol s hlavičkou pracovať, som si musel vytvoriť pomocnú metódu, ktorá získala všetky hlavičky tabuľky, porovnala ich názov s názvom hlavičky, ktorý potrebujeme a vrátila jej pozíciu, aby sme korektne potom mohli používať filter v danom stĺpci. Automatizácia tejto úlohy mi trvala okolo 6 hodín.

```
public void FilterByColumn(string column, string text)
{
    var input = _driver.FindElement(By.CssSelector($"tr.filters > th:nth-child({
        GetColumnIdx(column)}) > input"));

    input.SendKeys(text);
}

public bool IsTableFiltered(string column, string text)
{
    return _driver.WaitFor((driver) =>
    {
        return driver
                .FindElements(By.CssSelector($"td:nth-child({GetColumnIdx(column)})"))
                .All(item => item.Text == text);
    }, null, true);
}
```

```
private int GetColumnIdx(string column)
{
    var headers = _driver.FindElements(By.CssSelector("thead > tr:nth-child(1) > th")).
        ToList();

    var idx = headers.FindIndex(header => header.Text == column);

    if (idx == -1)
    {
        throw new ApplicationException("Column doesn't exist!");
    }

    return ++idx;
}
```

Výpis 4: Metódy na interakciu s filtrami

4.4 Testovanie pridávania škôl, kurzov, učiteľov a študentov cez import

Mojou úlohou ako tester bolo skontrolovať, či sa importujú všetky dáta korektne, pokiaľ dáta obsahujú nesprávané údaje (napríklad telefónne číslo ma nesprávny formát alebo chýba), tak boli nahradené výplňovým textom, ktorý označoval, že dáta neboli správne. Úloha bola rozdelená na podúlohy tak, že každý vývojár a tester zaobstarával časť importu. Mne bolo priradené testovanie importu škôl a učiteľov. Testovanie tejto úlohy mi zabralo približne 5 hodín.

Začal som zostavením testovacích scenárom pre import škôl a učiteľov. Pre import dát z Cleveru som používal skript v Postmane[7] [9], ktorý vykonal sériu HTTP volaní, ktorými importoval okres s daným identifikátorom do testovacieho prostredia Pegwriting. Pre množstvo škôl, ktoré bolo treba skontrolovať, bolo nutné overiť, či sa správne importovali nielen v užívateľskom rozhraní, ale aj priamo v databáze.

Pre import škôl som si zvolil okres, ktorý ponúka Clever na testovanie, lebo obsahuje najväčšie množstvo hraničných prípadov, ako sú chýbajú údaje, emailové adresy v nesprávnom formáte atď. Spustil som import podľa identifikátora daného okresu. Vytvoril som viacero SQL skriptov, ktorými som zistil počet škôl a učiteľov, ktorí boli importovaní, ako aj priradenie učiteľov školám. Následne som tieto hodnoty porovnal s hodnotami v prieskumníku dát v Cleveru. Pri prvotnom importe bol problém, že niektorí učitelia neboli importovaný kvôli chýbajúcim dátam a dátam, ktoré boli pridlhé na limity nastavené v databáze a záznamom, ktoré sú vyžadované a chýbajú v dátach od Cleveru. Riešením bolo použitie výplňových dát pre chýbajúce polia a pre dáta, ktoré mali nesprávny formát. Po následnom nasadení a importe dát, už boli importované všetky školy a učitelia. Následne som v užívateľskom rozhraní aplikácie overil, že sa dané školy zobrazujú správne a funkcia vyhľadávania, triedenia škôl a užívateľov funguje správne.

[7] Postman je nástroj na zostavovanie a testovanie API aplikácií, vykonávanie HTTP dotazov a pod.

```
SELECT s.name, COUNT(DISTINCT u.UserID) as 'Teachers'
FROM [User] u
JOIN [UserSchool] us ON u.UserID = us.UserID
    AND role = 'proctor'
JOIN [School] s ON s.SchoolID = us.SchoolID
    AND s.DistrictID = 358
GROUP BY s.SchoolID, s.name
```

Výpis 5: Priradenie učiteľov školám

Tabuľka 2: Výsledok priradenia učiteľov školám

Škola	Počet učiteľov
City High School	41
Pineapple Elementary School	28
Rockaway Beach Middle School	22
Rockaway Beach Middle School Campus 2	2
Rockaway Beach Middle School Campus 3	3

```
SELECT COUNT(*) as 'Teachers count'
FROM [User] u
JOIN [UserSchool] us ON u.UserID = us.UserID
    AND role = 'proctor'
JOIN [School] s ON s.SchoolID = us.SchoolID
    AND s.DistrictID = 358
```

Výpis 6: Počet učiteľov v danom okrese

4.5 Testovanie pridávania študentov do kurzu a udelenie licencií počas importu

Na to, aby som správne otestoval danú funkcionalitu, som sa musel najprv zoznámiť so systémom ako fungujú licencie v rámci Pegwriting.

Licencie sa udeľujú na úrovni škôl v okrese. Študenti, ktorí sú priradení k škole, ale nie sú zaradení v žiadnom kurze nevyužívajú licencie. Licencia je priradená študentovi, až keď je zaradený do kurzu. Študent zaradený vo viacerých kurzoch spotrebúva vždy len 1 licenciu. Licencie priraďujú učitelia, buď už vloženým študentom, alebo vložením nového študenta do systému.

Po tom, ako som si naštudoval ako sa pracuje s licenciami, som začal s vytváraním testovacích scenárov. Ak má pri importe dát škola dostupné licencie, tak budú tieto licencie priradené importovaným študentom. Pokiaľ vezmeme do úvahy počet licencií, ktoré môže mať škola pri importe, môžu nastať tieto prípady:

- **Škola nemá dostupné licencie** - žiaden študent nebude priradený do kurzu.
- **Škola nemá dostatočné množstvo licencii** - študenti, na ktorých už neostane licencia, nebudú pridaní.
- **Škola má toľko licencii, koľko bude importovaných študentov** - všetci študenti budú priradení, škole neostanú licencie.
- **Škola má viacej licencii ako počet importovaných študentov** - všetci študenti budú priradení, škola bude mať ďalšie dostupné licencie.

Keď vezmeme do úvahy vzťah študenta ku kurzu, máme tieto scenáre:

- **Študent nie je priradený do kurzu** - študent nebude zaberať licenciu.
- **Študent je priradený práve do jedného kurzu** - študent bude využívať 1 licenciu.
- **Študent je priradený do viacero kurzov** - študent bude využívať 1 licenciu.

Importovanie dát z Cleveru môže byť vykonané viacnásobne, a preto môžu nastať tieto prípady:

- **Študent bol odstránený z kurzu v Pegwriting** - pri importe sa študent priradí kurzu a bude zaberať licenciu.
- **Študent bol odstránený z kurzu v Cleveru a má len 1 kurz** - pri importe sa študent odstráni z kurzu a uvoľní licenciu.
- **Študent bol odstránený z kurzu v Cleveru a má viacero kurzov** - pri importe sa študent odstráni z kurzu.
- **Študent bol priradený na iný kurz v Pegwriting** - pri importe sa študent preradí na predošlý kurz, počet licencií sa nezmení.
- **Študent bol priradený na iný kurz v Cleveru** - pri importe sa študent preradí na nový kurz, počet licencií sa nezmení.

Po vytvorení testovacích scenárov na základe týchto prípadov, som vytvoril nový okres v Cleveru. Následne som vytvoril CSV súbory obsahujúce 5 škôl spolu s kurzami a študentami, ktoré zahŕňali dané prípady, ktoré som následne preniesol na Clever, aby som ich mohol importovať do Pegwriting-u.

Po implementácii som začal s testovaním. Pri prvom importe, v ktorom sa pridáva celý okres, školy nemajú priradené licencie a tak podľa očakávania neboli priradení žiadni študenti do kurzov a import obsahoval správy, ktoré túto udalosť oznámili. Následne som školám priradil licencie podľa tabuľky 3. Po opätovnom importovaní sa korektne priradili študenti do kurzov. Pri *SDE Clever School 2* sa správne importovalo len prvých 5 študentov a ďalší už neboli priradení kvôli chýbajúcim licenciám. Po preradení/odstránení študentov z kurzov a dokončení ďalšieho importu

sa študenti korektne priradili späť k pôvodným kurzom. Posledným krokom bolo odstránenie študenta z Clever portálu a nasledný import. Po overení tohto scenára, som zapísal krátky popis priebehu testovania k danej user story a uzavrel som ju. Otestovanie implementácie spolu s vytvorením testovacieho prípadu mi trvalo približne 10 hodín.

Tabuľka 3: Počet licencií a študentov škôl

Škola	**Počet študentov**	**Počet licencií**
SDE Clever School 1	5	5
SDE Clever School 2	10	5
SDE Clever School 3	5	10
SDE Clever School 4	1	2
SDE Clever School 5	5	5

5 Teoretické a praktické znalosti uplatnené v priebehu praxe

V priebehu praxe som využíval množstvo znalostí, ktoré som nadobudol počas štúdia na Vysokej škole báňskej – Technickej univerzite Ostrava. Rád by som vyzdvihol niektoré z nich.

Veľmi užitočný bol obecný prehľad o vývoji softvéru, dôležitosti častej iterácie, ako aj o výhodách a významu použitia návrhových vzorov, o ktorých som sa dozvedel v predmetoch Vývoj informačných systémov a Úvod do softvérového inžinierstva. Pri automatizácii testov som využil znalosti o platforme .NET a jazyku C# získané v predmetoch Programovacie jazyky II a Architektúra technológie .NET.

Pri práci s databázou som využil znalosti z predmetov ako sú Úvod do databázových systémov a Databázové a informačné systémy, v ktorých by som vyzdvihol dôraz na vytváranie správnych a efektívnych SQL dotazov. Tieto znalosti som využil hlavne pri testovaní integrácie s Cleverom, kde bolo nutné prechádzať veľké množstvo záznamov a bolo nutné, aby bol daný SQL dotaz čo najefektívnejší.

Pri testovaní administratívneho portálu som využil znalosti o webových technológiách z predmetu Vývoj Internetových Aplikácií. Využil som hlavne znalosti o CSS pri testovaní dizajnu daných komponentov, ako aj znalosti o zostavovaní xPath dotazov a selektorov pre DOM elementy, ktoré som využil pri automatizácii.

6 Znalosti a zručnosti chýbajúce v počas praxe

Počas vykonávania odbornej praxe mi chýbali znalosti hlavne z oblasti testovania softvéru, ktorá sa na bakalárskom stupni nevyučuje. Určite by bolo prínosné pre študentov bakalárskeho stupňa mať predmet, ktorý sa zameriava práve na túto problematiku, lebo si myslím, že tieto znalosti sú užitočné aj pri samotnom vývoji softvéru. V praxi sa často človek stretáva s tým, že je nutné tvoriť testy, či už na úrovni unit testov alebo až na úrovni akceptačných testov.

Ďalšou znalosťou, ktorá mi pri praxi chýbala, boli vedomosti o verzovacích systémoch, ktoré sú nevyhnuté pri vývoji akéhokoľvek softvéru. Aj pri práci testera som narazil na situácie, v ktorých som potreboval vytvoriť vetvu z hlavnej vetvy repozitára pri implementácií jednotlivých automatizovaných testov a následne tieto zmeny propagovať späť.

7 Záver

Vykonávanie bakalárskej práce formou odbornej praxe vo firme hodnotím ako výbornú príležitosť, ako sa naučiť pracovať v tíme, rozšíriť si znalosti zo školy, ako aj nadobudnúť vedomosti, ktoré nie sú obsahom výuky.

Počas praxe som mal možnosť využiť viacero prístupov k testovaniu aplikácií, pri ktorých som využil a prehĺbil znalosti nielen z oblasti testovania softvéru, ale aj zo samotného vývoja. Prax ma naučila dôležitosť správneho použitia návrhových vzorov, ako aj písanie čitateľného a udržateľného kódu, lebo sa pri vývoji softvérového produktu často stáva, že stávajúca funkcionalita sa musí rozšíriť o novú, a nečitateľný a nesprávne navrhnutý kód znemožňuje jednoduché rozšírenie tejto funkcionality. V priebehu praxe som sa zoznámil s procesným frameworkom scrum, ktorý je v súčasnosti najvyužívanejším agilným prístupom k vývoji softvéru.

Veľkou výhodou vykonávania odbornej praxe vo firme SDE Software Solutions hodnotím spoluprácu s americkými firmami. Pri práci na projekte je väčšina komunikácie vedená v anglickom jazyku, ktorého znalosti som si počas praxe vo veľkej miere rozšíril.

Literatura

[1] *Selenium Documentation* [Online] [cit. 2018-03-20]
Dostupné z: `https://www.seleniumhq.org/docs/`

[2] *Specflow Documentation* [Online] [cit. 2018-03-19]
Dostupné z: `http://specflow.org/documentation/`

[3] *Gherkin Documentation* [Online] [cit. 2018-03-20]
Dostupné z: `https://github.com/cucumber/cucumber/wiki/Gherkin`

[4] Dima Kovalenko - *Selenium Design Patterns and Best Practices*, Birmingham: Packt, 2014, 270 s. ISBN 9781783982707

[5] *JavaScript End to End Testing Framework — Cypress.io* [Online] [cit. 2018-03-20]
Dostupné z: `https://www.cypress.io/`

[6] *WebdriverIO - WebDriver bindings for Node.js* [Online] [cit. 2018-03-20]
Dostupné z: `http://webdriver.io/`

[7] *Powering technology in the classroom — Clever* [Online] [cit. 2018-04-17]
Dostupné z: `https://clever.com/`

[8] *Browserstack - Cross Browser Testing Tool* [Online] [cit. 2018-04-17]
Dostupné z: `https://www.browserstack.com/`

[9] *Postman — API Development Environment* [Online] [cit. 2018-04-17]
Dostupné z: `https://www.getpostman.com/`